

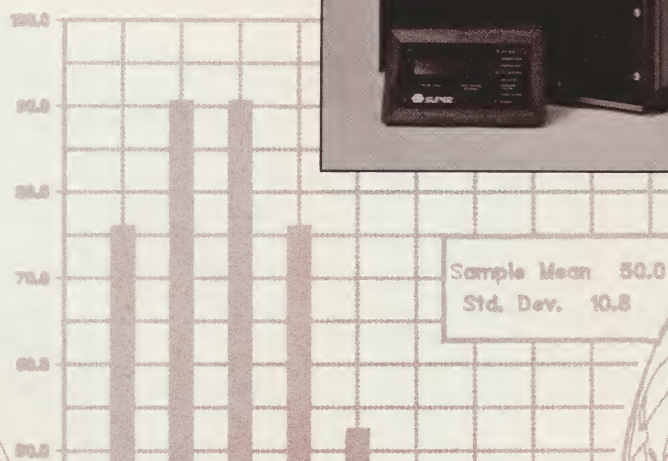
★ May 80

PGM

48 Bit Virtual Memory Mainframe Power Plus DISSPLA[®] Graphics



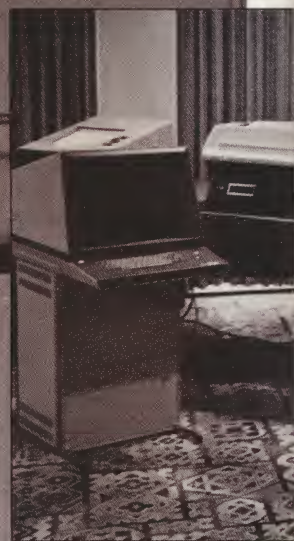
ONIC ORTHOMORPHIC
JECTION



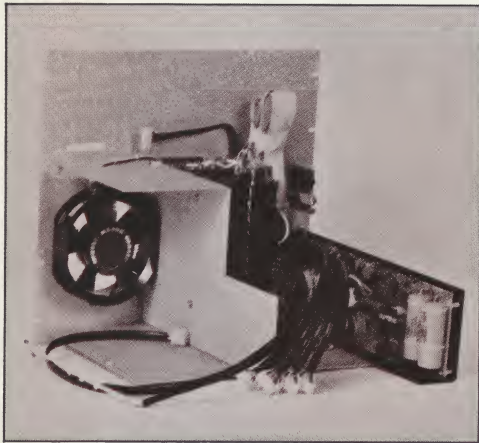
This plot illustrates the use of two Blank areas simultaneously with Bars Story and Dotted Grid

 **SUPERSET** inc

 **SUPERSET** INC.



PGM — GRAPHICS BASED DISTRIBUTED PROCESSOR, CONTROLLER OR STAND-ALONE COMPUTER



The PGM is a cost effective innovative device for implementing massive computer programs. Originally designed principally to transfer the overhead of the DISSPLA[®] graphics system from large mainframes, its 48-bit precision and Virtual Memory have proven capable of implementing megabyte size programs beyond the reach of minicomputers. Its minicomputer price makes it an ideal device for either a dedicated controller or interactive workstation.

The entire DISSPLA graphics package is included at no extra charge.

DISSPLA is by far the leading graphics software available and produces publication quality charts, graphs, maps and drawings. It is device independent and is implemented on about 300 mainframes world wide.

Features include:

- Over 60 quality character sets, including shaded fonts,
- 3-D hidden line surfaces with axes and projected text,
- 14 world map data bases and 16 map projections,
- Shaded areas and bars with arbitrary spaced patterns,
- Log, linear, month labeled, user labeled, date labeled and degree/minute axes with grids,
- Parametric spline and polynomial interpolation plus smoothing,
- Right and left justified text and legends,
- Dot, dash, chaindot and arbitrary pattern lines.

These are just a sampling of DISSPLA features. *The price of DISSPLA alone is comparable to that of the entire PGM.*

An Intelligent Distributed Processor

Unlike most "intelligent terminals," the PGM does not merely provide an RS232 plug for host computer communication. The PGM provides software and hardware intelligence to ensure data integrity, enable error correction and account for arbitrary non-data messages. Software is provided for the host computer to append flags, counts and check sums to the data. The PGM recognizes this data; strips off the appended flags, etc.; *makes sure that all data lines are in correct sequence* and none are missing; and *compares the sum check on*

every line. Missing lines and those with sum check errors are noted. Software is provided to retransmit arbitrary lines and merge them into a previously transmitted block.

Host data is automatically buffered to disk. Extraneous received text, such as host operator messages, is immediately displayed on the console terminal. The user can enter "dumb terminal mode" at any time and communicate with the host as a TTY terminal. Thus operator communication, system commands and other host functions can be carried out without interfering with data transmission.

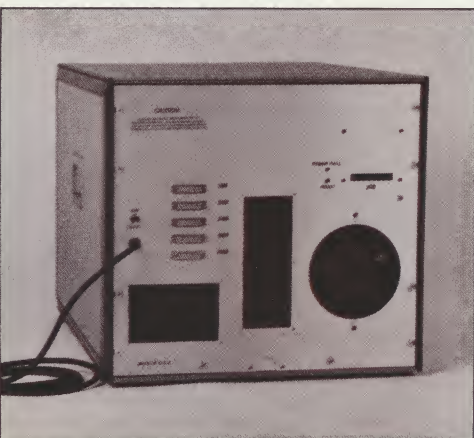
The PGM acknowledges the existence of other computers and embraces them intelligently without any modifications to their systems.

A Fast Interactive Workstation

The PGM provides five (5) RS232 ports, each of which can be set for 110-38,400 baud communication. These ports are totally device independent, allowing the user to assemble a workstation to suit individual preferences and needs. An optional interface is provided for a 9 track tape drive for disk backup; making tapes for plotters, microfilm units, etc.; and transferring software.

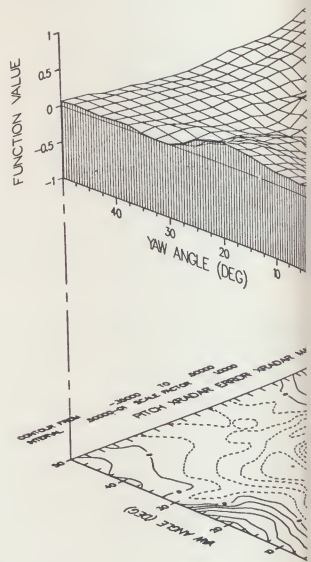
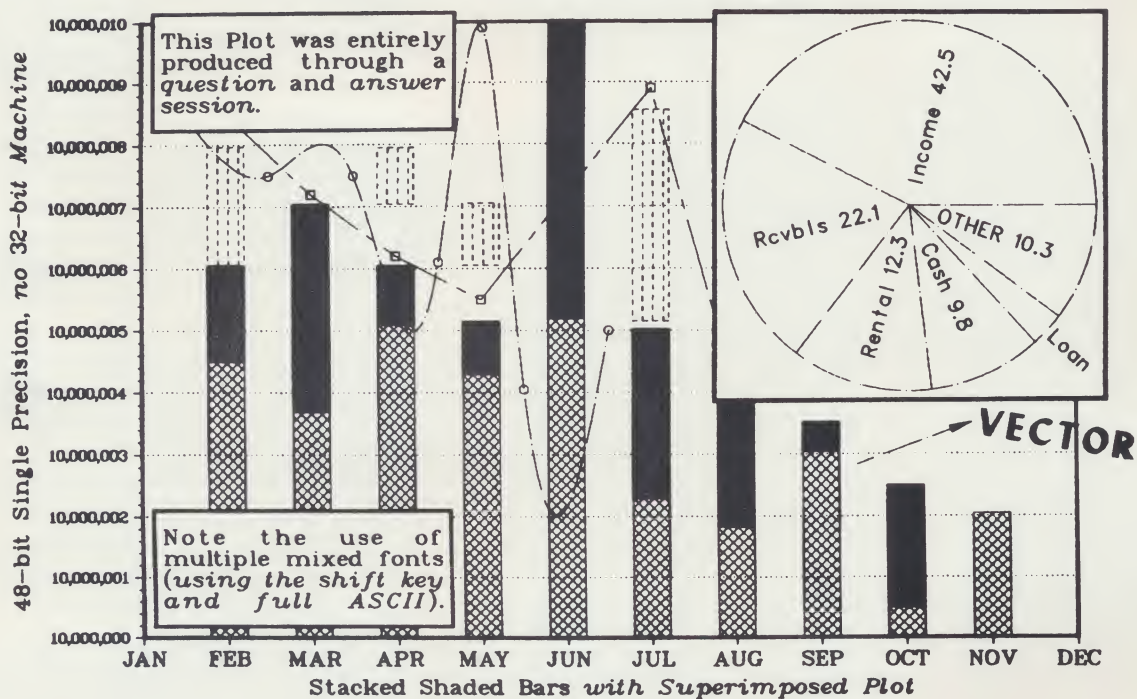
A typical workstation might consist of an inexpensive console CRT terminal (Hazeltine, Lear Siegler, etc.), a color monitor (Chromatics, ISC, etc.), a high resolution monochromatic terminal (Tektronix, Megatek, etc.), a plotter for hardcopy (Zeta, Houston Instruments, Calcomp, etc.) and a modem for host communication. Any such combination or subset is acceptable. Thus the user can interact through the scrolling CRT console, observe the relationship of the graphics colors on an inexpensive color monitor, study the *identical* graphics on the high resolution Tektronix device and produce permanent hardcopy of the final graphics on the plotter. The final graphics can also be transferred to a tape for a Dicomed microfilm unit, Xynetics flatbed, Zeta drum plotter, etc.

PGM units can be networked to communicate with one another. The PGM is a remarkably versatile device. It does not commit one to any brand of peripheral hardware, nor to any configuration.



Non-Prog. I/Active DISSPLA

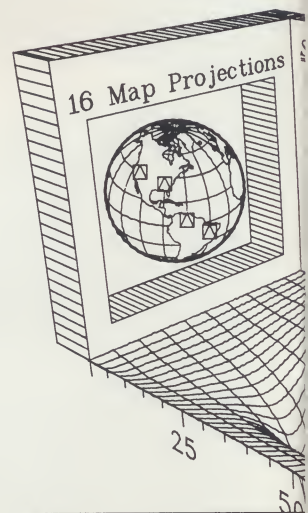
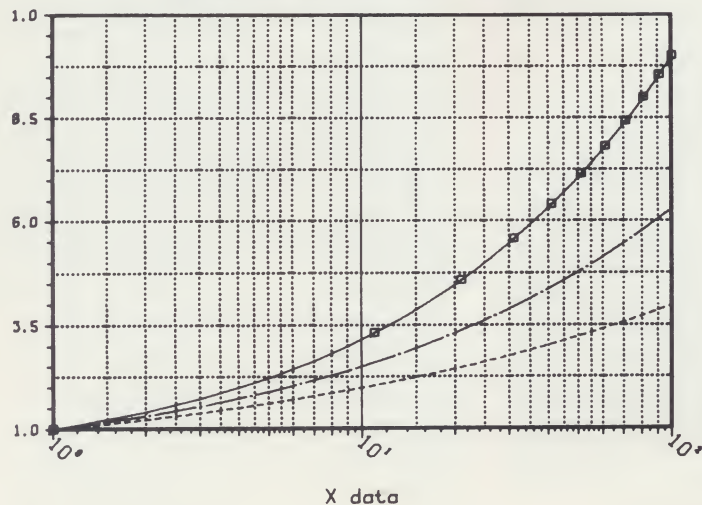
Plot Produced Entirely on a P.C.M.



TOFF EQUAL AREA PROJECTION

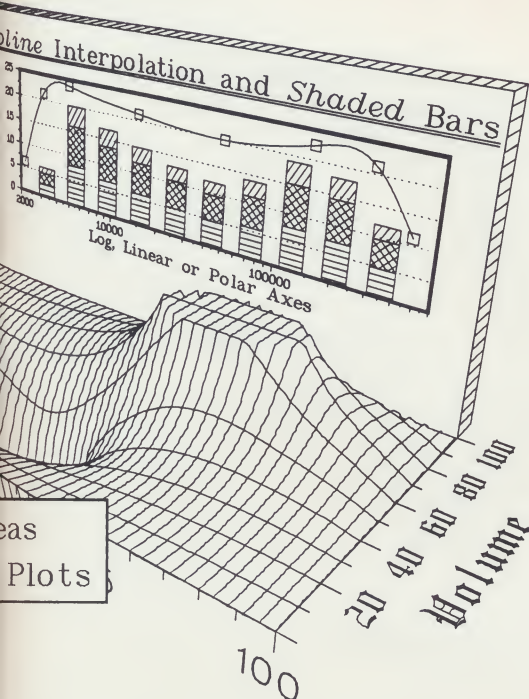
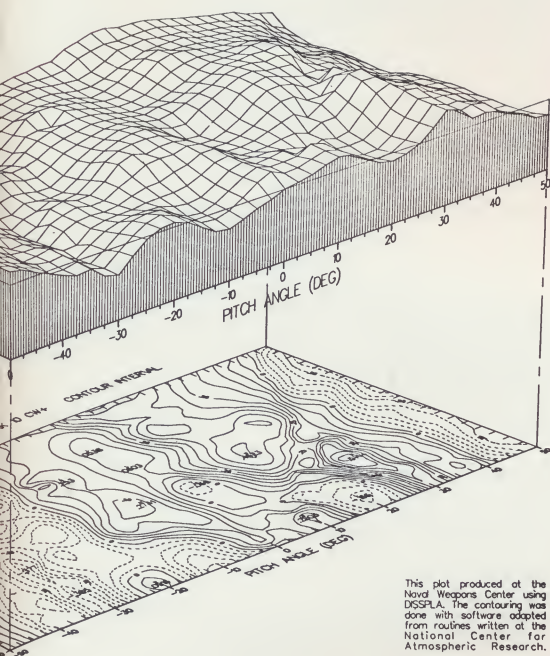


DOTTED SEMI-LOG GRID



Blanked Ar
Projected 2-D

Maxi Software, Mini Box



The true power of the PGM is its software. The PGM offers a system and spectrum of programs unheard of in any minicomputer. Few multimillion dollar maxis can offer the following (at no extra charge):

- **DISSPLA.** The unchallenged king of graphics software.
- **INTERACT.** Non-programmer oriented frontend to DISSPLA. Offers most features found in TEL-A-GRAF®, except that it is interactive. Interact is based on menu selection and question/answer rather than a special language. Interact produces its graphics concurrently with the user question/answers; there is no pseudo-batch waiting for plots, or "trial and edit." This *true interactive* implementation of DISSPLA speeds the production of publication quality charts and slides by orders of magnitude. Interact offers *interactive map drawing and projection* in addition to bar charts, pie graphs, log plots, etc. *in full color.*
- **POSTGRAPH.** All graphics can be stored in *device independent* "files." Postgraph will selectively retrieve and draw specified plots. It can scale, rotate, segment, window and superimpose an arbitrary sequence of plots from one, or more, files. For example the titles can be taken from one file, the axes from another and the data curves from yet another. Plots can be scaled down and superimposed on blanked areas in other plots. Plots can be enlarged and scissored into a sequence of device size tiles, automatically. *The output can be produced on an arbitrary set of graphics devices simultaneously.*
- **Host DISSPLA Shell.** A surrogate DISSPLA subroutine library for a host computer. The Shell furnishes all the subroutine commands of DISSPLA, but does not produce any plots. It records the DISSPLA calls and transmits them to a remote PGM. The PGM actually executes the graphics. This cuts the DISSPLA memory overhead on the host by a factor of up to 6:1 and the host execution overhead up to 100:1. *The Shell furnishes DISSPLA capability to programs on PDP 11's and other minis, in addition to slashing DISSPLA overhead on maxis. Source code is furnished for custom user modifications.*
- **Virtual Memory Operating System.** The PGM provides a highly interactive

sophisticated maxicomputer operating system. Output can be directed to files and multiple devices simultaneously. Programs can be run from within programs e.g., the compiler (or any other processor) can be run from within the editor, which in turn can be creating a data element from the middle of Interact and so on. Data output files can be redirected and assigned from the middle of the editor or any other program. Date, time and programmer I.D. are attached to elements and load modules. Directories can be scanned with the editor and ordered alphabetically or chronologically. Programs can be halted at any time, the output files and memory contents examined interactively.

- **Fortran SUPERSET Compiler.** The compiler will not only *accept* ANSI Fortran, but is vastly extended to incorporate *structured features of PASCAL and ALGOL.* SUPERSET provides IF... THEN... ELSE, DO... END DO, FOR... STEP... UNTIL/WHILE, WHILE loops, CASE, internal PROCEDURES with LOCAL variables, ENCODE/DECODE, Boolean bit operators, Field manipulation, Byte/Character manipulation, execution time FORMAT field definition, in-line comments and many other features. *Names can be up to 30 characters long and upper/lower case used arbitrarily.* Loops can have real variables and expressions as limits. All local variable values are preserved from one subroutine execution to the next. Extensive cross-reference tables can be produced if desired.
- **Editor With Typesetting.** The editor "lines" can be up to 3000 characters long. These paragraphs can be folded into arbitrary width columns with right and left justification. Masks can be set for locating within, and the printing of, specified fields, defined by column number and/or string literal bounds e.g., string matching and printing can be restricted to the fields 7-40, 45-'ABC' and ';'-'') concurrently. Arbitrary text blocks can be moved around duplicated, inserted and *transferred from other elements.* Column alignment can be preserved during text modification.
- **Multi Level File Management.** Data is organized into a natural hierarchy of

elements, nodes, files and silos.

Subroutines and other distinct data items are stored as elements. Elements are organized in nodes such that *several versions of the same element can appear under the identical name in different nodes*. Libraries of nodes constitute files and file editions can be backed up on tape silos. Elements can be transferred between nodes, deleted and copied. *The contents of all elements are sum checked on all transfers and backups to ensure data integrity*. The file management system performs *dynamic garbage collection* i.e., deleted disk space is recycled immediately. Text and program load images are compressed. The PGM optimizes use of disk space more than almost every other computer *at any price*.

- **Linker With Cross-Reference Tables.**

The Link Editor can be directed to selectively scan specified node libraries. Key subroutines can be grouped for virtual memory execution efficiency. Checks are made for recursive subroutine call sequences and the full call sequence for all unresolved items is printed. COMMON block lengths are checked for mismatch. Pages are packed as densely as possible. *Elaborate cross-reference tables of all entry points and COMMON blocks*, dates, times and programmer ID of compilations, page contents and page fill statistics are available.

Large Scale Software Processor

The PGM is dedicated to the processing of massive programs in an interactive personal environment. The PGM is able to link, load and execute the *entire* DISSPLA package as a single load image — *almost 2 megabytes of mostly instructions*. This feat is beyond the limited addressing capabilities of many of the multimillion dollar mainframes. The very fact that the PGM was capable of compiling and *linking* the 40,000 odd Fortran DISSPLA statements, divided over about 350 subroutines, shows that it is capable of handling programs beyond most minicomputers. DISSPLA has proven too much for the limited capabilities of these machines.

The PGM is in the price class of a minimal 16 bit mini, but its capabilities place it in the league of the maxi mainframes.

48-Bits — Almost A Double Precision VAX

The PGM leapfrogs 32 bit machines with its 48-bit precision. The 11 + digit floating point precision of the PGM is almost double the 5-6 digits of 32 bit computers. Pumping a few sines, logs and exponentials repeatedly through a 32 bit machine *reduces its reproducible accuracy to about 4 digits*. That is about a foot in a mile — hardly acceptable to a surveyor, astronomer, cartographer or highway engineer. Four digits is roughly a hundredth of an inch in a foot — hardly acceptable to a machinist or mechanical engineer.

The PGM has a floating point precision of a cent in a billion dollars.

One can always use double precision on a 32 bit machine. This, however, tends to slow programs *by up to an order of magnitude*. One also has to put up with the transfer between integers, single and double precision — as seasoned programmers are well aware. DISSPLA knows nothing about double precision, which can be a problem if one wishes to draw a map in standard meter or foot coordinates on a 32 bit machine.

Forty-eight (48) is divisible by 3, 4, 6, 8, 12, 16 and 24. The PGM is at home in Octal, Hexadecimal, full ACSII and 6 bit character codes. It interfaces 8 bit micros, 16 bit minis and 12 bit A/D converters perfectly.

Virtual Memory Without Virtual Memory

THE PGM gives the programmer the illusion of virtual memory — one can define many huge arrays and link hundreds upon hundreds of subroutines into a single load module. The PGM is to all intents and purposes a "Virtual Memory" machine with a *24 megabyte address space*.

But the PGM does not employ cache memory, bank translation registers, hardware page translation tables and other costly devices to implement Virtual Memory. Yet the PGM averages *over 70,000 Fortran statements per second* with no more hardware than a basic 16-bit mini. The PGM employs an intelligent approach to implementing the higher level language software.

Most computers are not designed to take advantage of the characteristics of higher level languages. The PGM takes full advantage of the fact that only about 10% of

the memory references, on the average, are array references, subroutine calls and other potential virtual addresses. The PGM employs high speed *direct addressing* — for loop indices, local variables, branch vectors, strings and other items which are found to account for about 90% of all memory references. So 90% of the time the PGM behaves as a simple, basic direct address machine needing very little memory hardware assist. For the other 10% it uses intelligent interaction between the hardware microcode and the software.

The PGM represents an intelligent marriage of hardware to higher level language software. *It is an elegant low entropy engineering solution.*

A True Extended Fortran Engine

The PGM has no programming registers, no status words, no accumulators and *no assembly language*. The PGM instruction set is a direct implementation of the output of its extended Fortran Compiler. The statements $A=B+C$, $I=J/K$, $Q(J)=P(K)*R(L)$ are all single instructions. This speeds the execution of higher level language programs substantially, by eliminating extraneous register diddling. It also compacts the program instruction occupancy by more than double compared to most other machines; the single PGM instruction $A(I)=B(J)*C(K)$ can require almost a dozen instructions to implement on most machines. This reduces the amount of page swapping by eliminating the pages in the first place — thereby speeding the throughput.

The instruction set is similar to the PASCAL P code idea, except that it is not stack based and is substantially closer to the original instructions.

Frontend Microprocessor Handles All I/O

The high speed TTL bit slice full 48-bit processor is concerned only with crunching programs, it passes all I/O tasks to an Intel 8080 microprocessor. The 8080 is responsible for interfacing all device handshaking, RS232, tape drive protocols, priority handling and other chores for which it is eminently suited. The 48-bit processor can go about its business without being concerned about the outside world, much as a CDC 6600. This not only speeds the overall execution time, but also simplifies

the entire machine design considerably. It also eliminates the possibility of the 48-bit processor "getting lost" due to a systems software bug, thereby simplifying the operating system and making it more robust.

Hardware Checks On Software Errors

The hardware microcode checks for the following software errors automatically:

- Array bounds faults — references to any array beyond its declared limits.
- Uninitialized variable references — attempt to use a variable never assigned a value.
- Subroutine arguments count errors — mismatch in count of arguments and formals lists.
- Unnormalized real numbers — attempt to use an integer value in a floating point operation (usually occurs when an integer is accidentally passed as a floating point subroutine argument).

The hardware microcode also conducts "Fuzz Factor" rounding of small floating point differences. For example the statement $IF(A=B)$ THEN is actually executed by the hardware as if it were: $IF(ABS(A-B) < FUZZ)$ THEN where *FUZZ* is a user presettable constant. Thus the PGM hardware can recognize that 0.9999... is equal to 1.0 whereas virtually all other machines will not. This rounding also applies to $A-B$, $A > B$, $A < B$, $A = B$ and $A \leq B$ operations.

These checks are so powerful for software development that the DISSPLA implementation on the PGM was checked out in less than a week — *an all time record*; in spite of the fact that the operating system was under development. *These checks picked up a number of bugs in DISSPLA itself that had yet to be discovered.*

Hardware Trace Buffer Gives Bug History

The PGM is equipped with a hardware buffer that records the last 64 operations in *real time*. It is always active, and its contents are available for review when a fault is detected. It can be set to trace all transactions, processor memory references only or branch operations only.

If a program error occurs, the 48-bit processor halts and the Intel 8080 can be

used to probe the contents of its memory or *even alter its contents*. The microprocessor can also be used to examine the trace buffer. Once the bug is found, *a correction can be patched in memory and execution resumed*. A "breakpoint" can also be set in the code to force a halt at a specified virtual address; *this is unusual for a Virtual Memory machine.*

The PGM has proved to be the most powerful software development tool *at any price*. Bugs that take weeks to find on many machines can usually be found in minutes, and the PGM detects bugs that have yet to be found on other machines.

A Portable Box

The entire PGM including its 29MB disk is contained in a microwave oven size cabinet. It is provided with handles for ease of portability and has been transported in everything from a Chevette hatchback to the back seat of a single engine light airplane. It consumes about 750 watts of 110 volt power i.e., less than most electric toasters and requires no raised floor, air conditioning or special power branch circuits. *It is ideal where mobility is important.*

PGM

PGM Specifications

Hardware Furnished As Standard

- Full 48-bit TTL bit slice software processor unit,
- Intel 8080 I/O frontend,
- Intel Multibus I/O interface,
- 29-MByte Winchester Technology disk,
- 48K words (288K bytes) error correcting memory, expandable,
- 5 RS232 ports 110-38,400 baud programmable,
- Detached panel with column count and software/hardware status readouts,
- Hardware trace buffer selectively records last 64 operations,
- All power supplies monitored for glitches exceeding $\pm 5\%$ of rated voltages,
- Automatic powerdown in case of overheat.

Optional Hardware

- 16K word (96K byte) error correcting memory boards,
- Industry standard "Pertec" 9 track tape drive interface,

Power Required — Standard 110 volt, approximately 750 watts.

Size — 19X18X25 inch microwave oven-size cabinet.

Air Conditioning — None.

Software Included With Standard PGM

- Full DISSPLA Graphics subroutine package,
- INTERACT interactive DISSPLA frontend for non-programmers,
- POSTGRAPH plot management postprocessor,
- DISSPLA Shell subroutine package for host computers,
- Host Computer error detection/correction communication software,
- Virtual Memory SUPERSET Operating System (SOS),
- Fortran compatible SUPERSET compiler with ALGOL/PASCAL structures,
- Editor with word processing and typesetting,
- Disk Management System with fault recovery and automatic dynamic garbage recycling,
- Multilevel File Management System for developing megabyte-size programs,
- Link Editor with Cross-Reference Tables,
- Trace debugging facility with editor to search for variables by name and value,

- PEEK debugging processor to interactively probe a virtual memory image by subroutine name,
- Tape Management System for backing up individual items or entire disk contents, with extensive sum checking.

People Behind the PGM

The PGM design team was led by Ian Hirschsohn. Ian was the originator and principal author of DISSPLA. He was the founder and first president of Integrated Software Systems Corporation (ISSCO). He was also the author of the AMESPLOT, DISSCUS and TURBSTAT large scale software packages.

Ian was backed by Brian Doore and Dick Stanford, two highly competent and experienced applications oriented systems programmers. Being a close knit highly productive software team they were able to accomplish a feat beyond the innumerable committees of large companies.

The PGM software is a product of their many years of experience with IBM 1620, 1401, 1130, 7094, 7044, 360 and 370; UNIVAC 1230, 1106 through 1100/81; CDC 3600, 6600, 7600 and Cyber; Burroughs B6700; Elliott 503; PDP 11; and MODCOMP computers. The PGM SUPERSET Operating System is a distillation of the ideas they extracted from this wealth of experience — it is a product more of hands-on experience than enthusiasm.

The hardware group consisted of Jim Hunter and Paul Wolf. They were founders of Digital Scientific Corporation; principal designers of the META 4 (used extensively as an IBM 1130/1800 emulator) and later the IBM 370/158 plug compatible, marketed as the IteL AS/5. They are a pair of the most respected designers of microprogrammable computers since the mid-1960's.

The PGM stands as testimony to Brooks' Law that the achievement of a team is inversely proportional to its size.



11675 Sorrento Valley Road, Suite L
San Diego, California 92121
Telephone (714) 452-8665

My 80

INTRODUCING

PGM

Distributed Processor, OEM Controller or
Standalone Computer with Interactive Graphics.

48 Bit Virtual Memory Mainframe Power plus DISSPLA[®] Graphics.

INCLUDES: Full 48 bit processor, Intel 8080 I/O frontend, 29MB disk, 300k bytes error correcting memory, 5 RS232 ports, Compiler, Editor, Linker, Virtual O/S-system, DISSPLA Subroutines and Interactive DISSPLA.

OEM systems already delivered.

The PGM is compiling, linking and executing megabyte size programs of over 40,000 Fortran lines and over 350 subroutines apiece. 11 + digit floating point far exceeds 32 bit machines.

PGM is the first true Fortran Engine. Fortran statements e.g., $A=B+C$, $Q(J)=P(I)/R(K)$ — executed as single 3 address instructions. Array bounds, uninitialized variables and illegal floating point values hardware checked on every operation. Cyclic hardware buffer records last 64 operations.

DISSPLA publication quality graphics presently on almost 300 mainframes, is included. Fortran Superset Compiler has structured features of PASCAL and ALGOL.

Standard Intel Multibus I/O provides interface flexibility. No air conditioning. Standard 110 volt plug. Self-contained, compact and portable microwave oven size cabinet.

DISSPLA is a registered trademark of Integrated Software Systems Corporation

For more information call or write:



11675 Sorrento Valley Road, Suite L
San Diego, CA 92121
(714) 452-8665

\$35,000 QUANTITY
OF 10

\$49,000 SINGLE
UNIT

